# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-207849
(43)Date of publication of application : 07.08.1998

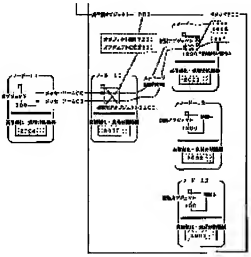
(51)Int.Cl. G06F 15/16
G06F 15/16
G06F 9/46

(21)Application number : 09-006576
(22)Date of filing : 17.01.1997

(71)Applicant : NIPPON TELEGR & TELEPH CORP <NTT>
(72)Inventor : TAKEMOTO MITSUHARU

## (54) HIGH RELIABILITY AND LOAD DISTRIBUTION METHOD FOR DISTRIBUTION SYSTEM

(57)Abstract:



PROBLEM TO BE SOLVED: To simultaneously execute fault detection, internal state coincidence and load distribution and to attain a high reliablity/load distribution mechanism capable of reducing the number of messages by sharing a time measurement start request message or the like for fault detection due to time-out, autonomously collecting information related to loads to a suitable node and using a clock of a false management object.

SOLUTION: The time measurement start request message for fault detection due to time out and a duplicate message for object restart at the time of a fault are shared. A message for maintaining the consistency of internal states of duplicates, a disuse request message for a duplicated message to be required at the time of a fault and the notification message of information related to loads to respective nodes 11 to 13 are shared. Thus the reexecution of a duplicate is prevented by autonomously collecting information related to loads to a node 10 and using the clock 1000 of the false management object of which existence is sensed from the external.

* NOTICES *

2.**** shows the word which can not be translated.

3.In the drawings, any words are not translated.

## CLAIMS

[Claim(s)]

[Claim 1]In a distributing system based on object-orientation, this object is reproduced to plurality, Make one of the reproduced objects into an execution system, make the remainder into a standby system, and high-reliability-izing and a load sharing mechanism are formed for every plat form of each node, Although substance as an object does not exist, it is that only a name of an object and an identifier exist in said high-reliability-izing and load sharing mechanism, It is set up so that a false management object observed as it exists from the outside may be installed regardless of a node in which this duplicate object exists, A high reliability object is formed by this false management object and said duplicate object, While storing a name of an object, and an object identifier and forming high-reliability-izing and a load sharing mechanism in a plat form of a node in which this false management object exists, It is observed by existence of an object from the exterior and this high-reliability-izing and load sharing mechanism, After detecting a message transmitted to this addressing to a false management object, transmit this message to said execution system duplicate OBUJIEKU, start a method of this duplicate object, perform operation of the purpose of this object, and at the time of an end of execution of a method. By transmitting a message which maintains the consistency of an internal state from an execution system duplicate object to a standby system duplicate object, High-reliability-izing and a load sharing mechanism on a node in which said false management object exists when the consistency of an internal state between duplicate objects is guaranteed and an execution system duplicate object breaks down, By cooperation with high-reliability-izing and a load sharing mechanism on a node in which a standby system duplicate object exists. Detect said failure, next choose one of the duplicate objects, make this duplicate object into an execution system duplicate object, and a message which an execution system duplicate object before failure received is reproduced, By transmitting this message to an execution system duplicate object after failure detection, An execution system object which recovered said failure and was newly chosen after detection of said failure, A rerun is barred when acting on external objects other than this object, and this external object is an operation of the 2nd henceforth, When failure does not occur in an execution system duplicate object, and cancelling unnecessary information in information about duplicate management for every end of 1 method execution of this execution system duplicate object and communicating information about duplicate management between nodes, High-reliability-izing and a load sharing method in a distributing system transmitting information about load of this node simultaneously, and performing load sharing using this information.

[Claim 2]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a message which maintains the consistency of an internal state of said duplicate object, As a checkpoint for every end of execution of one method of an execution system duplicate object, Information about an internal state of an execution system duplicate object is notified to high-reliability-izing and a load sharing

mechanism of a node in which a standby system duplicate object exists, High-reliability-izing and a load sharing method in a distributing system, wherein said high-reliability-izing and load sharing mechanism in which this information was received update an internal state of a standby system duplicate object.

[Claim 3]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a procedure of said failure detection, A message addressed to this false management object to which high-reliability-izing and a load sharing mechanism of a node in which said false management object exists have been transmitted from the outside is detected, Transmit this message to a duplicate object of an execution system, and this execution system duplicate object is made into a method run state, Transmit a duplicate of this message to high-reliability-izing and a load sharing mechanism of a node in which a standby system duplicate object exists, simultaneously, and high-reliability-izing and a load sharing mechanism in which this message was received carry out an opportunity [ reception of this message ], the node, when information about an internal state transmitted to within a time [ which started time measurement and was beforehand defined with a local timer ] at a node of a standby system duplicate object at the time of an end of execution of a method of an execution system does not arrive, High-reliability-izing and a load sharing method in a distributing system judging that an execution system duplicate object is out of order.

[Claim 4]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a procedure of said failure recovery, When the operation according to claim 3 detects failure, in order to change an execution system, A priority which makes an execution system duplicate object the highest priority is given to all the duplicate objects in order, Inside of high-reliability-izing and a load sharing mechanism of a node in which a standby system duplicate object exists, A duplicate object which exists in this node and has the next priority of an execution system is made into an execution system, From high-reliability-izing and a load sharing mechanism of a node in which a false management object exists. A duplicate of a message thought to have been indicated to claim 3 is transmitted to a duplicate object which was a standby system of this node, High-reliability-izing and a load sharing mechanism of a node in which other duplicate objects exist, and high-reliability-izing and a load sharing mechanism of a node which a false management object exists are received, . Transmit a message which shows that an execution system changed and have by high-reliability-izing and a load sharing mechanism of a node in which this false management object exists. High-reliability-izing and a load sharing mechanism on a node in which a duplicate object which changed information which has managed which duplicate object an execution system is, and newly changed to an execution system exists, As opposed to high-reliability-izing and a load sharing mechanism on a node in which a duplicate object which was an execution system before exists, High-reliability-izing and a load sharing mechanism of a node in which a message of a reboot of an object or a node is transmitted and what has a priority high next exists in a duplicate object, High-reliability-izing and a load sharing method in a distributing system changing to an execution system and performing the above-mentioned recovery instead when it is not reported that an execution system changed to within a time [ which was defined beforehand ] after failure detection.

[Claim 5]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a procedure which prevents an operation 2nd after an object of said exterior, Although it is a different duplicate object, at the time of failure recovery in one high reliability object, In order to prevent influence on the exterior by the same method being performed, high-reliability-izing and a load sharing mechanism of a

node in which the false management object according to claim 3 exists, it can set to the node when a message is received from an object identifier and the exterior of a false management object in a message which transmits to a duplicate object — physical or logical time [ being local ], [ attach and ] When an execution system duplicate object gives an operation to the exterior by processing requested from message transmission to the exterior, or an execution environment, a node to which an object identifier of said false management object and a false management object exist in an argument of a function of processing requested from this message or an execution environment — local time, [ give and ] Defined these as a rerun prevention tag collectively, and received a message with this rerun prevention tag, or when there is a request of processing with a tag, make the end product of this tag and processing by them into a group, and high-reliability-izing and a load sharing mechanism save, When there was the same thing as this tag at the time of a rerun and it is detected, Without rerunning when the end product is required, High-reliability-izing and a load sharing method in a distributing system suppressing side effects of a rerun of a duplicate object and considering a rollback point which should be chosen from a checkpoint as an end time of the latest method by that cause by returning a saved value.

[Claim 6]A procedure which cancels what becomes unnecessary for information about said duplicate management in high-reliability-izing and a load sharing method in the distributing system according to claim 1, In order to cancel what became unnecessary for information which manages a duplicate, at the time of an end of execution of the existing method according to claim 2. When information about an internal state is notified to a standby system duplicate object from a duplicate object of an execution system, about message transmission to the exterior and execution environment request processing which were performed within the method according to claim 5, When high-reliability-izing and a load sharing mechanism of a node in which a duplicate object of an execution system exists transmit to high-reliability-izing and a load sharing mechanism of a near node which acted, a message which cancels information saved with a tag, High-reliability-izing and a load sharing method in a distributing system cancelling information which became unnecessary.

[Claim 7]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, in said high-reliability-izing and load sharing mechanism. A message handling mechanism in a kernel has a small quantity and a high-speed translation table which are changed into an identifier of a duplicate object from an identifier of a false object, Transmit a message to a duplicate object of an execution system, and an end of a method of a duplicate object of an execution system of a high reliability object is detected, In a node which notifies an end of method execution to a duplicate message management server object of a node in which a standby system duplicate object exists and in which a duplicate object of a standby system exists in a duplicate message management server object, If a notice in which a duplicate of a message which a duplicate object of an execution system received was saved at, and a duplicate object of an execution system ended execution of a method trouble-free is received, By not coming to within a time [ which a terminating notice of this execution defined beforehand further ], at the time of failure detection, cancel a saved message, detect failure of an execution system, issue directions of recovery also to other nodes further, and a duplicate object identifier management server object, When tables changed into an identifier of a duplicate object from an identifier of a false object in a kernel run short, by the inquiry from a message handling mechanism in a kernel. Return an identifier of a duplicate object and an object executed result

management server object, When a certain object ends execution of a method, preservation as a result of a message used as an opportunity of starting of this method and execution is performed, At the time of failure, it detects that it is a message retransmission from the same high reliability object, Prevent performing the same operation and an object duplicate object execution control server object, High-reliability-izing and a load sharing method in a distributing system managing information about a state of execution of a duplicate object on a certain node, or a priority of high-reliability-izing, and performing cancellation and starting of a duplicate object.

[Claim 8]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a procedure of load sharing between said nodes, As opposed to high-reliability-izing and a load sharing mechanism of a node in which the standby system duplicate object according to claim 2 exists, When notifying information about an internal state of an execution system duplicate object, High-reliability-izing and a load sharing mechanism of a node in which information showing load of a node in which an execution system duplicate object exists is notified, and a standby system duplicate object exists, When a direction of a self-node detects that load is light using said information, High-reliability-izing and a load sharing mechanism of a node in which a switching request of an execution system is notified to high-reliability-izing and a load sharing mechanism of a node in which a false management object exists, and this false management object exists, High-reliability-izing and a load sharing method in a distributing system choosing which [ of the duplicate objects ] is made into an execution system using information about notified load.

[Claim 9]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a procedure of load sharing between said nodes, By notifying information other than load information for every time of a node to high-reliability-izing and a load sharing mechanism in a node in which a false management object exists, High-reliability-izing and a load sharing method in a distributing system choosing which [ of the duplicate objects ] is made into an execution system, and performing load sharing dynamically.

[Claim 10]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a procedure of load sharing between said nodes, High-reliability-izing and a load sharing method in a distributing system characterized by moving a duplicate object to a node with light load using information on load of a node accumulated on a node in which a false management object indicated to claim 8 exists.

[Claim 11]In high-reliability-izing and a load sharing method in the distributing system according to claim 1, a procedure of load sharing between said nodes, High-reliability-izing and a load sharing method in a distributing system characterized by deciding a generation place and a priority of a duplicate object using information on load of a node accumulated on a node in which a false management object indicated to claim 8 exists.

## DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention]In this invention, two or more execution substance is divided into an execution system and a standby system in the object which operates by a distributed computing environment. Therefore, it is related with the recording medium which recorded high-reliability-izing in the distributing

system which can realize high-reliability-izing and load sharing, the load sharing method, and its processing program.
[0002]
[Description of the Prior Art]There is art of operating an object by a distributed computing environment, by making the object which is a motion unit of an object-oriented system into the method of high-reliability-izing. When a distributing system realizes high-reliability using a duplicate, the overhead of the mechanism in which consistency is maintained between the duplicate becomes large. For example, in the method of using the below-mentioned PassiveReplication. All the duplicates process in the same order also by the method of the special processing for maintaining internal state consistency being needed, and using Active Replication, and processing in which the operation to the exterior is summarized to one is needed. With the distributing system, since it is difficult to detect failure efficiently, in many systems, failure has been detected using a clock. However, although failure has not arisen, when time to exceed anticipation for the load of a communications network or the node itself is spent on processing, by this method, it may be regarded as failure. In an international congress "InternationalSymposium on Fault-Tolerant Computing" as an example actually studied conventionally, "The Delta-4 Approach to Dependability in Open Distributed Computing System" (1988), There is the architecture called Delta-4 announced by "Using Passive Replication in Delta-4 to Provide Dependable Distributed Computing" (1989). Although this is studied also about which [ of PassiveReplication/ActiveReplication ] case, the above-mentioned problem about failure detection is not still solved.
[0003]There is Fault-Tolerant Concurrent C by Active Replication as an example of the approach which supports the processing which summarizes the operation to the exterior to one by the linguage processing system. This to "International Symposium on Fault-Tolerant Computing." It is announced as "Fault-Tolerant ConcurrentC:A Tool for Writing Fault-TolerantDistributed Programs." Since the system call in which a kernel adjusts the operation to the exterior of two or more duplicates would also be added automatically and will have realized Active Replication if an execution code is obtained using this linguage processing system, an application programmer is not made conscious of existence of a duplicate. However, since the overhead at the time of execution becomes very large, when you do not need Active Replication, it cannot apply. As research using Passive Replication, There is "Implementing Fault-Tolerant Distributed Objects" published by "IEEE Transactions on Software Engineering" (the June, 1985 item). This is Passive Replication fundamentally and considers the mechanism in which the operation to the exterior is not made to perform. However, although it distinguishes that it is the operation from a certain Fault-Tolerant Distributed Object, the identifier of the execution thread is used, and it becomes a high cost in order to detect, if this becomes multistage. Usually, at the time of execution, without creating the duplicate of execution substance, only the data requirement is reproduced to the node and the system of nOR based on the idea of Lasy Fault Tolerance which becomes at the time of failure and reproduces execution substance is also studied. This is announced as "a high reliability-ized method of the distributed processing system based on data-decentralization-izing and object reconstruction" of the "Information Processing Society of Japan paper magazine" (the October, 1995 item). However, if it is this method, for this reason, time from failure recovery cannot adopt it as a system with a severe demand of real-time requirement greatly.
[0004]

[Problem(s) to be Solved by the Invention]Thus, in the above-mentioned all directions type, failure detection and consistency maintenance processing of the internal state are performed separately fundamentally. In this, between execution substance etc., an excessive message will fly about and degradation of performance will be caused. When creating two or more execution substance, load sharing can also be performed besides high-reliability-izing, but for load sharing, the information on the load of each node is needed. In order to collect these information, a new message flies about and it becomes one cause of performance degradation. After all, it becomes a technical problem of this invention to solve the following problem. Namely, when creating the object which provides two or more execution substance and carries out high reliability operation in a Prior art on a distributing system, Since it has two or more points that excessive execution cost starts, and execution substance in losing the operation to the exterior by the reboot of an object which is chosen and changed and in which the consistency maintenance of the internal state during a duplicate, failure detection, and an execution system carry out high reliability operation, It is the point that excessive execution cost also starts realization of the load sharing using it. The purpose of this invention solves the technical problem of these conventional technologies, and can perform simultaneously failure detection, internal state coincidence, and load sharing, Also when high-reliability-izing and a load sharing mechanism with few messages can be realized and two or more high reliability objects are performed simultaneously, it is in providing high-reliability-izing and the load sharing method in the distributing system of low cost used as hindrance.

[0005]

[Means for Solving the Problem]In order to attain the above-mentioned purpose, in high-reliability-izing and a load sharing method in a distributing system of this invention. A time test starting request message for failure detection by timeout, Carry out the duplicate messages for an object reboot at the time of failure in common, and A message for consistency maintenance of an internal state during a duplicate, A cancellation request message of the duplicate messages which must have been needed at the time of failure, By carrying out a notification message of information about load of each node which is needed for load sharing in common, bringing information about load together in a suitable node autonomously, and using outside a clock of a false management object by which existence is sensed, The reruns of a duplicate are prevented and this obtains high reliability execution and load sharing execution. A storage which recorded a processing program for gaining high reliability execution and load sharing execution is realized.

[0006]

[Embodiment of the Invention]Hereafter, a drawing explains the principle of operation and example of this invention in detail.

(Principle of operation) Drawing 9 is a figure of the switchboard control by the object of two or more nodes to which this invention is applied. In recent years, in order to process more correctly early the information which complicates [ extensive-ization ] and broadens and is generated with diversification of a socioeconomic activity, and an advancement, the importance of the information processing system using a computer is increasing. For example, in a digital data exchange, a message-exchange function is needed in the ability of many users to communicate with many partners. Since a variety of computers and terminals are connected to a digital data exchange, the information processing function corresponding to them is also needed. In drawing 9, in the data switching network, two or more nodes are connected by the

communications network (switchboard), and the driver object and IN object are installed within each node. Transmission and reception of a message are performed between the objects in each node. Drawing 10 is a figure of a distributed plat form (distributing system). When message communication is being performed between the objects 108 of two or more nodes 25, in the portion above the plat form 3000, he does not need to be conscious of a distributing system at all. On the other hand, in the portion below the plat form 3000, i.e., servers, and the kernel, he is conscious of the distributing system. Between kernels, actual communication is performed via communications network hardware.

[0007]By the way, methods for realizing high reliability operation of such an information processing system include the technique of generally reproducing execution substance. There is Passive Replication in which the remainder stands by by only Active Replication in which all the duplicates perform the target function simultaneously as an execution gestalt of this duplicate, and one duplicate performing the target function. Although recovery of overly high-speed failure is possible for Active Replication, it is necessary to collect what acts a duplicate outside, and execution cost becomes high. On the other hand, although recovery of the failure in which Passive Replication is overly high-speed is difficult compared with Active Replication, it is suitable for performing with a distributing system, and execution cost can also be held down low. Although high reliability-ization is gained by Passive Replication in a distributed object inclination execution environment, it is necessary to guarantee the consistency of an internal state between duplicates (the following, duplicate object). Unless it guarantees this, right execution cannot be guaranteed when execution substance (the following, execution system) is changed. In order to guarantee this, an internal state is saved in the place which is hard to be destroyed in the timing of a suitable checkpoint. There is no inconvenience also on the calculator memory which the preservation place in this case has in a remote place also at a place like the neighborhood of CPU which is operating, for example, a hard disk. When failure breaks out, the optimal point (rollback point) will be determined between related objects from a set of the time saved, respectively, and execution will be resumed from there. Since an object may act outside via message transmission etc. mutually, generally it is difficult to determine a rollback point from this checkpoint. This has description, for example in Yoshifumi Manabe of Information Processing Society of Japan "information processing" (the November, 1993 item), and Shigeki Aoyagi collaboration "distributed checkpoint rollback algorithm."

[0008]Drawing 7 is a figure showing the situation at the time of failure breaking out, when there is no duplicate of a message, drawing 3 is a figure showing reception of the message of a high reliability object, and drawing 5 is a figure showing the end of the method of a high reliability object. In this invention, the influence on the exterior by a message etc. is solved by considering it as the mechanism which prevents a rerun by making the end time of each method under execution of all the high reliability objects into a checkpoint for the aforementioned problem. In rebooting an object, the message used as the opportunity of execution of an object is needed. When it was a method, i.e., the method which transmits the message 420 only to the duplicate object 105 of an execution system, as shown in drawing 7 and the node 20 whole breaks down, the reboot will be impossible, even if it is lost by message 420 the very thing and an execution system changes to the duplicate object 105. Namely, although the node 20 has the execution system duplicate object 105 and the standby system duplicate object 106 exists in the node 21, It becomes the situation which cannot be rerun since there is no message 420 even if it is lost by the message 420 in failure

and changes to the node 21, since only the object 105 transmitted the message 420. Therefore, the duplicate of a message as shown in drawing 3 is needed. When there will be no answer by the time beyond fixed time passes using a clock in order to perform failure detection with a distributing system, it is common to use the timeout method suspected to be failure. However, at this time, it is necessary to determine, respectively whether to measure time until measuring time from which time or what happen.

[0009]In this invention, it has solved combining the duplicate of the message which mentioned this problem above, and the message for a consistency guarantee of an internal state. That is, as shown in drawing 3, the reproduced message is distributed by each node (11, 12, 13), this starts measurement of time by the nodes 12 and 13 of a standby system, and time until the message (412,413 of drawing 5) for a consistency guarantee of an internal state comes at the time of the end of a method is timed. Thereby, the excessive message which is needed only for failure detection does not fly between nodes. The object identifier 201 of the local time of a false management system (node 10 with the false management object 1100 of drawing 3) and the false management object 1100 is used, The operation performed via a message etc. out of the duplicate object of the execution system in a high reliability object is identified. The influence which the same message from the same high reliability object by the reboot of a duplicate object has outside by this can be lost. It can be coped with also when there is the moment two or more duplicate objects of an execution system exist at the time of failure, by this. And by notifying the loaded condition of the node (node 11 of drawing 5) of an execution system to the node (similarly nodes 12 and 13) of a standby system into the message of an internal state consistency guarantee, When the node of a standby system judges autonomously and judges that the duplicate object on a self-node should become an execution system, it notifies to the node (the node 10) of a false management system, and this also realizes load sharing simultaneously. Drawing 3 and drawing 5 are explained further in full detail.

[0010](EXAMPLE)

[Composition of a high reliability object] Drawing 1 is a software configuration figure of the high reliability-ized system of an object in which one example of this invention is shown. In drawing 1, the high reliability object 1000 comprises the duplicate object 1200 of an execution system, and the duplicate object 1300-1301 of the standby system. For example, if the object identifier 201 of the external object of 100 shots and the message 400 addressed to name 301 of an object reach the node 10, High-reliability-izing and the load sharing mechanism of the node 10 (message handling function in a kernel) detect the message, and transmits to the duplicate object 1200. The duplicate object 1200 which received transmission of the message 400 performs a method. Since the message which transmitted is processed without also causing error [ what ] if it observes from the outside at this time, it is observed as the object of the object identifier 201 and the name 301 of an object exists. And existence of a duplicate object is not observed. On the contrary, the message from the duplicate object 1200 of an execution system is observed as if it was the message 401 transmitted by processing of high-reliability-izing and a load sharing mechanism from the object of the object identifier 201 on the node 10, and the name 301 of an object. Also at this time, from the exterior, it is observed as the object of the object identifier 201 and the name 301 of an object exists. According to that is, the processor of the message of high-reliability-izing and the load sharing mechanism of the nodes 10 and 12. The false management object 1100 which does not exist actually can exist, and the false management object 1100 can grasp that operation of the duplicate objects 1200-1300 and 1301 is

managed. It is observed as access from the outside is performed to the false management object 1100.

[0011][Composition of high-reliability-izing and a load sharing mechanism] Drawing 2 is a software configuration figure of high-reliability-izing and a load sharing mechanism. The details of each server object are mentioned later. In drawing 2, high-reliability-izing and the load sharing mechanism 2010 are stored in the plat form 3001 of the node 16, and this is reproduced, transmitted and stored in the nodes 17, 18, and 19 (respectively 2011-2012, 2013). The detection function of the message about an object with the kernel 2100 specific to the kernel of the usual distributed operating system is added. Although the usual inter object communication 422 is performed between the objects of the node 17 and the node 18, after getting down in the plat form 3001 actually, it can be distributed to the purpose object (423,424,425,426). When straddling a node, in straddling the node 19, for example, it goes via a message queue (2050-2060). Therefore, this detection function is realized by adding a little reconstruction to the scheduler message handler of the kernel 2100, etc. By this detection function, the operation to the message addressed to a false management object and the exterior of a duplicate object is detected, and processing related to high reliability operation and load sharing operation is performed. When a message is transmitted to the duplicate object of ****** from a false management object by the kernel, the duplicate of the message is made. This reproduced message is received and saved by the duplicate message management server object 2200 of the node in which the duplicate object of a standby system exists. The duplicate message management server object 2200 times time from this message reception, and is related also to failure detection.

[0012]When the message transmission from a different duplicate object in the same high reliability object, etc. break out, the object executed result management server object 2300 realizes the mechanism which prevents method starting by the message transmission. When a kernel obtains the identifier of the duplicate object of an execution system, etc. from the object identifier of a false management object, etc., the duplicate object identifier management server object 2400 is used in order to make small the memory area which a kernel uses. When there is no target entry in the translation table in the kernel 2100, specifically, it asks the duplicate object identifier management server object 2400. The contents of the translation table in a kernel can also be changed by sending a message to the duplicate object identifier management server object 2400 from the exterior. The duplicate object execution control server object 2500 manages the run state of the duplicate object which exists on the node. By the inquiry from the outside, a run state is returned or ** and the rebooting function also have an object at last. Since it is decided that the above-mentioned server object (2200-2500) will be each one node, an object identifier assigns the identifier easily understood from a node number. When performing below-mentioned high reliability operation and load sharing operation by carrying out like this, operation of obtaining the identifier of an object from the name of an object becomes unnecessary.

[0013][Message reception of a high reliability object] Drawing 3 is a figure showing the state where a high reliability object receives a message and starts a method. When a high reliability object is observed from the outside, it seems to exist in the place where a false management object exists as shown in drawing 1. If the message 405 from the outside reaches the node 10 in which the false management object 1100 exists, it will detect that high-reliability-izing and the load sharing mechanism 2000 (message handling mechanism in a kernel) of the node 10 are addressing to the false management object 1100. On the table which it had in the kernel, the duplicate object 1200 of an execution system, the duplicate object 1300-1301 of a standby

system, and each priority are acquired from the false management object 1100. When the data for which it asks on the table does not exist, it asks the duplicate object identifier management server object of high-reliability-izing and the load sharing mechanism 2000 in the node 10, and required data is obtained. Here, the priority of a duplicate object can operate this invention, if it gives in arbitrary order, but if a high priority is given to the duplicate object on the node 10 in which the false management object 1100 exists, and the node which can perform message communication early, it can raise performance. A difference arises in the earliness of the message communication between nodes because a channel is congested, condition also differs between each node and the loads of the node itself also differ. High-reliability-izing and the load sharing mechanism 2000 (message handling function in a kernel) transmit the received message 405 to the duplicate object 1200 of an execution system (406). Simultaneously, also to high-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) of the nodes 12 and 13 in which the duplicate object of a standby system exists, a message is reproduced and it transmits (407,408). The time of reception of the message 405 measured with the local timer of the node 10 is put into the reproduced next message (407,408). This time turns into method starting time of the high reliability object 1000. This time may also be the physical time which shows the time of the real world, or the logical time in the inside of a computer system. As logical time in a distributing system, it contains in a message and Vector Clock which makes what added one to the larger one among its logic time and the logic time in a message new logic time is mentioned as an example at the time of message reception. In the nodes 12 and 13 in which the duplicate object of a standby system exists, since high-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) save the message 407,408 which arrived, the duplicate object 1300-1301 does not operate. if high-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) receive the message 407,408 — a node — time measurement is started with a local timer. This is used for the failure detection mentioned later.

[0014][Execution of the method of a high reliability object] Drawing 4 is a figure showing the state of execution of a high reliability object. The duplicate object 1200 of an execution system operates as an object usual in a duplicate object in addition to the operation to the exterior. Transmission of a message is used when the object 1200 acts to the object exterior. Changing and rerunning an execution system at the time of a failure occurrence consists of a head of a method. Therefore, if the message transmission 410 to the external object 101 occurs [ a certain method 500 ], when a duplicate object will be rerun at the time of failure, the message of the same contents as the message 410 will be transmitted to the object 101. In this, it will secede from a target right state by ****** which receives a message with an excessive state of the object 101. For example, when the object 101 changes by the number of the messages which received the internal state, or when generating a new object outside, the side effects by failure retransmitting a message to an unnecessary message must be suppressed. That is, even if the high reliability object 1000 may perform the method 500 twice or more at the time of failure, the mechanism in which the operation to the external object 101 is not caused is required. From the execution system duplicate object 1200 of the high reliability object 1000 to the message 410. The value of the sequence counter about the operation to the exterior within the object identifier of a high reliability object, the message reception time in a false management object, a method identifier, and its method is given as the tag 1500.

[0015]Although the tag is attached also to the message which is unrelated to usual high reliability operation and load sharing operation, it only exists as the field in a message, and is not effective as a value. When a certain node 15 receives the message 410 addressed to object 101 of a self-node, it is investigated whether the tag 1500 about the high reliability is effective. If effective, it will be investigated whether there is anything of the same contents as the tag 1500. If there is nothing, it will be saved and execution of the method 501 will be started. And when execution of the method 501 is completed, the end product and tag 1500 are set and saved. When the thing of the contents as the tag 1500 with the effectively same tag 1500 exists, it is judged as that from which the execution system duplicate object changed by failure in the same high reliability object 1000, and execution of the method 501 is not performed. When it is the message transmission with an answer which returns the saved end product to a high reliability object, the saved end product is considered as an answer. When these saved information should be cancelled, it is at the end time of execution of the method of a high reliability object, and this is mentioned later.

[0016][An end of execution of the method of a high reliability object] Drawing 5 is a figure showing the end operation of execution of the method of a high reliability object. High-reliability-izing and the load sharing mechanism of the node detect the end of execution of the method 500 of the execution system duplicate object 1200, and the end is notified to high-reliability-izing and the load sharing mechanism 2001. It can see for having ended execution of the method 500 of the high reliability object 1000 from the external application object at this time. High-reliability-izing and the load sharing mechanism 2001 which received the terminating notice, If the message transmission to the exterior performed by the method 500 or the history of execution environment request processing is investigated and it exists, The above-mentioned tag 1500 and the message 411 of cancellation of holds information are transmitted to high-reliability-izing and the load sharing mechanism 2005 of the transmission destination node 15 (object executed result management server object). In high-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) of the nodes 12 and 13 in which the duplicate object of a standby system exists. The message 412,413 including the cancellation demand of the message (407,408 of drawing 3) used as the opportunity which starts the method 500, the information about the load of the node 10, and the information about the internal state of the execution system duplicate object 1200 is transmitted.

[0017]The node's 15 reception of the message 411 will investigate whether the tag 1500 about high-reliability-izing currently attached to it is effective. It is effective, and if the contents are cancellation of holds information, the executed result of the method corresponding to the tag will be canceled. If high-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) in the nodes 12 and 13 receive the message 412,413, The message corresponding to the method 500 is canceled out of the message (407,408 of drawing 3) transmitted from high-reliability-izing and the load sharing mechanism 2000 (ring functions, such as a message of a kernel) of the node 10 in which the false management object 1100 exists. The internal state of the duplicate object 1300-1301 of a standby system is updated for the information about the received internal state. Compare the information on load and the information on the load of one's node about the node 11 of an execution system duplicate object, and if the direction of its node has light load, The information on the load is notified to high-reliability-izing and the load sharing mechanism 2000 (duplicate object identifier management server object) of the node 10 in which a false management object exists.

[0018][Failure detection of a duplicate object] Drawing 6 is a figure showing the failure detection which used timeout. The message 405 transmitted to the high reliability object 1000 from the exterior, High-reliability-izing and the load sharing mechanism 2000 (message handling function of a kernel) of a false management object detect actually that it is addressing to the high reliability object 1000, and The duplicate object 1200 of an execution system, It transmits to high-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) of the nodes 12 and 13 in which the duplicate object 1300-1301 of a standby system exists. High-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) of the nodes 12 and 13 store the message 407,408. simultaneous -- the node in a kernel -- a time test is started with a local timer. When execution of the method 500 of the execution system duplicate object 1200 is completed correctly, the message 412,413 to that effect should come to high-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) of the nodes 12 and 13 on the contrary. If this is not completed to within a time [ which was decided beforehand ], it judges with failure.

[0019][A change of the duplicate object by failure detection] High-reliability-izing and the load sharing mechanism 2002-2003 (duplicate message management server object) in which failure was detected as mentioned above, The priority of the duplicate object 1300-1301 which is operating on its node is investigated, and if its direction is high, it will be judged that the duplicate object which is operating on its node becomes an execution system. Here, the direction of the duplicate object 1300 presupposes that a priority is high. The duplicate object 1300 which newly became an execution system receives the message 407 saved from high-reliability-izing and the load sharing mechanism 2002 (duplicate message management server object) of its node, and performs a method. High-reliability-izing and the load sharing mechanism 2002 (duplicate object execution control server object) raise one priority of the duplicate object 1300. Next, high-reliability-izing and the load sharing mechanism 2002 (duplicate message management server object), High-reliability-izing and the load sharing mechanism 2000 (duplicate object identifier management server object) of the node 10 with the false management object 1100, and high-reliability-izing and the load sharing mechanism 2003 of the node 13 with the duplicate object of other standby systems. The change rate of an execution system is notified to a (duplicate object identifier management server object). High-reliability-izing and the load sharing mechanism 2003 (duplicate object execution control server object) raise one priority of the duplicate object 1301 on their node.

[0020]High-reliability-izing and the load sharing mechanism 2001 (duplicate object execution control server object) of a node in which the duplicate object of the execution system existed before are received, High-reliability-izing and the load sharing mechanism 2002 (duplicate object execution control server object) notify the demand of a duplicate object reboot. A notice of this message will reboot the duplicate object 1200 of the node 11. If the duplicate object 1200 still exists at this time, it will be rebooted after cancellation. When a duplicate object is rebooted by the node 11, the internal state is not maintaining consistency. However, when execution of a certain method is completed, the duplicate object of an execution system transmits the message which guarantees the consistency of an internal state to the duplicate object of a standby system. Therefore, even if the duplicate object reboot of a standby system occurs at the time of failure, the processing which guarantees the consistency of an internal state is not newly needed. The reaction of the duplicate object 1200 is [ even if ] slow by the load of the node 11 being

unusually high etc., In spite of not being out of order, even if it is judged that it is out of order, the above-mentioned recovery progresses and the duplicate object 1200 of the original execution system and the duplicate object 1300 of a new execution system exist, the influence on the exterior does not come out by operation of an object executed result management server object. After high-reliability-izing and the load sharing mechanism 2003 (duplicate message management server object) of the node 13 with a low-priority standby system duplicate object detect failure, if the change rate of an execution system does not arise within fixed time, It judges that all the standby system duplicate objects whose priority is higher than itself are out of order, and the same processing as the above for becoming an execution system by oneself is performed. When renewal of these tables is performed, the contents of the data it became impossible to put on the table in a kernel are notified to a duplicate object identifier management server object.

[0021][A change by the load of a duplicate object] Previously, explanation of drawing 5 showed the state of the end of execution of the method 500 in the high reliability object 1000. Then, when the duplicate object of an execution system ends a method, The mechanism which the information on the load of each nodes 11, 12, and 13 accumulates on high-reliability-izing and the load sharing mechanism 2000 (duplicate object identifier management server object) by operation of high-reliability-izing and the load sharing mechanisms 2001-2002 and 2003 was explained. By this high-reliability-izing and the load sharing mechanism 2000 (duplicate object identifier management server object) of the node 10 in which a false management object exists, Which duplicate object distinguishes whether it performs by the node with light load, and determines which duplicate object should be made an execution system on and after next time. Thereby, high-reliability-izing and load sharing are simultaneously realizable.

[0022][A change by the external information of a duplicate object] As mentioned above, the selection method of an execution system can be changed by giving the information on node selection to not only the load information of the node at the time of execution of the last method but high-reliability-izing and a load sharing mechanism 2000 (duplicate object identifier management server object) directly from the exterior. Specifically, the table about the duplicate object in the kernel in high-reliability-izing and the load sharing mechanism 2000 and the table which a duplicate object identifier management server object has are changed by transmitting a message to a duplicate object identifier management server object. Thereby, even if it does not use execution of a method, a more flexible execution system can be chosen.

[0023][Load sharing when movement of an object is possible] Drawing 8 is a lineblock diagram of the load distribution system in the case of moving an object in an object-oriented distributing system. In drawing 8, one high reliability object is constituted from the false management object 1100, the execution system duplicate object 1200, and the standby system duplicate object 1300-1301, Simultaneously, one high reliability object consists of the false management object 1101, the execution system duplicate object 1201, and the standby system duplicate object 1302-1303. The information on the load of the nodes 11, 12, and 13 and the information on the load of the nodes 22, 23, and 24 are accumulated by high-reliability-izing and the load sharing mechanism 2000 (duplicate object identifier management server object) of the node 10. Let that external information be the information about this accumulated load in the load distribution system which gives the external information in this invention (refer to claim 9). For example, even if the load of the nodes 22, 23, and 24 becomes high and it is trouble-free, processing of the execution system duplicate

object 1201 becomes slow, When are diagnosed as failure by the node 23 and 24 grades, and the load of the nodes 11, 12, and 13 is low, it is also possible to move the duplicate objects 1201−1302 and 1303 to the nodes 11, 12, and 13, and to raise the whole throughput.

[0024][A utilizing method of the accumulated load information] By this invention, to the node in which a false management object exists. Since the information about the load of a peripheral node is accumulated without performing message communication only for collecting especially the loads of a node, When generating a high reliability object, it is possible to apply to judgment to what node a duplicate object should be generated and whether what we do with the priority of a duplicate object.

[0025][Application to Call Manager] In addition, the object called Call Manager which exists in the call processing program in a distributing system is also an applied object. Call Manager will generate Callee which performs call processing by the side of Caller which actually performs call processing, or a receptacle, if a certain member's call origination is detected. Since it has two or more execution substance and processing can be requested appropriately at the light place of load, the whole throughput goes up. In the example of load sharing, the following things become possible by sharing the method (refer to claim 8) by the load of a node, and the method (refer to claim 9) by external information to selection of an execution system. Namely, it is a broader−based distributing system, and since it can be experientially predicted for example, when the load of the node in a specific geographical place increases according to the time difference in the human world, operation system should just give the information toward a high reliability object. If it does in this way, the CPU resources of a free node can be borrowed in time in a certain area, and it is possible to raise the whole throughput.

[0026][Application to high−reliability−izing and load sharing of a name server] As other applied objects, high−reliability−izing and load sharing of the name server (server which manages the name of an object and the identifier of an object) in a distributing system occur. A name server is a server indispensable although an object communicates on a distributing system. Since it is concealed thoroughly from what duplicate object the client which comes for the inquiry about a name is requested actually and high−reliability−izing and load sharing nature are also realized if this invention constitutes a name server, an aspect of practical use is also satisfactory.

[0027][The recording medium which records high−reliability−izing of a distributing system, and the processing program of the load sharing method] High−reliability−izing and the load sharing method based on the object−orientation stated above are realizable as a processing program for performing high−reliability−izing and load sharing, and the processing program can be recorded on a recording medium, and can be provided.

[0028]

[Effect of the Invention]As explained above, according to this invention, it becomes realizable [ high−reliability−izing and a load sharing mechanism with few messages which perform simultaneously failure detection, internal state coincidence, and load sharing ]. And properly speaking, the message of failure detection flies about in parallel to execution of the usual object, but the message preservation by another node and the message for internal state coincidence perform failure detection. It is also possible to add load information to the number of messages for internal state coincidence, and to perform load sharing. By these, the number of messages is greatly reducible. In this invention, since a multiple server realizes

high-reliability-izing and a load sharing mechanism, also when performing two or more high reliability objects simultaneously, it does not become hindrance. According to this invention, unnecessary waiting can be deleted with the mechanism in which the operation to the exterior of a high reliability object is suppressed, also about the mechanism of failure detection and recovery. That is, if it is the failure detection by the usual timeout, if there is no answer into fixed time, it will suspect for failure, and processing of a check of whether to be really failure is needed. In this invention, if there is no answer into fixed time, another execution system can be started promptly and deletion of a front execution system will be requested. It is uninfluential even if some front execution systems are acting outside simultaneously, before deletion is performed.

[Brief Description of the Drawings]

[Drawing 1]It is a figure of the high reliability object which makes a reserve system object reproduction which shows one example of this invention.

[Drawing 2]It is the lineblock diagram which realized high-reliability-izing and the load sharing mechanism in this invention by the kernel and the server object.

[Drawing 3]It is a figure showing reception of the message of the high reliability object in this invention.

[Drawing 4]It is a figure showing execution of the method of the high reliability object in this invention.

[Drawing 5]It is a figure showing the end of the method of the high reliability object in this invention.

[Drawing 6]It is a figure showing the failure detection of the duplicate object in this invention.

[Drawing 7]It is a figure showing the generation state and recovery impossible situation of failure in the former.

[Drawing 8]It is an explanatory view of the load sharing by object movement in this invention.

[Drawing 9]It is a figure of the switchboard control by the object of two or more nodes to which this invention is applied.

[Drawing 10]It is a figure of the distributed plat form where this invention is applied.

[Description of Notations]

10,11,12, 13,14,15, 16,17,18, 19,20,21, 22,23,24 — Node, 100, 101, 102,103,105,106 — Object (general), 201 [ — Method, ] — An object identifier, 301 — The name of an object, 400, 401, 422, 423, 424, 425, 426, 405, 406, 407, 408, 410, 411, 412, 413,414,415,420 — A message, 500,501 1000 — A high reliability object, 1100-1101 — A false management object, 1200-1201 — Execution system duplicate object, 1300-1301, 1302, 1303 — Standby system duplicate object, 1500 — A tag, 2000-2001, 2002, 2003, 2004, 2005, 2006, 2010, 2011, 2012, 2013 — High-reliability-izing and a load sharing mechanism, 2050-2060 — A message queue, 2100 — The message handling mechanism in a kernel, 2200 [ — Plat form. ] — A duplicate message management server object, 2300 — An object executed result management server object, 2400 — A duplicate object identifier management server object, 3001

---

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

**特開平10－207849**

(43)公開日　平成10年(1998)８月７日

| (51)Int.Cl.[6] | 識別記号 | ＦＩ | |
|---|---|---|---|
| Ｇ０６Ｆ　15/16 | ３７０ | Ｇ０６Ｆ　15/16 | ３７０Ｎ |
| | ４７０ | | ４７０Ｂ |
| 9/46 | ３６０ | 9/46 | ３６０Ｂ |

審査請求　未請求　請求項の数11　ＯＬ　（全 17 頁）

(21)出願番号　特願平9－6576

(22)出願日　平成９年(1997)１月17日

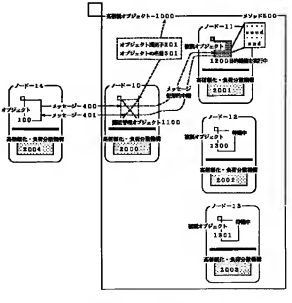
(71)出願人　000004226
日本電信電話株式会社
東京都新宿区西新宿三丁目19番２号

(72)発明者　武本　充治
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内

(74)代理人　弁理士　磯村　雅俊　（外１名）

(54)【発明の名称】　分散システムにおける高信頼化と負荷分散方法

(57)【要約】
【課題】故障検出と内部状態一致と負荷分散を同時に行うことが可能な、メッセージ数の少ない高信頼化・負荷分散方法を実現する。
【解決手段】タイムアウトによる故障検出のための時間測定開始要求メッセージと故障時のオブジェクト再起動のための複製メッセージを共通にし、複製間の内部状態の一貫性維持のためのメッセージと故障時には必要になるはずであった複製メッセージの破棄要求メッセージと負荷分散に必要になる各ノードの負荷に関する情報の通知メッセージを共通にし、負荷に関する情報を自律的に適切なノードに集め、外部には存在が感じられる擬似管理機構の時計を用いることにより、複製の再実行を防ぐことで、高信頼実行と負荷分散実行を得る。




本複製オブジェクトの構造

1

【特許請求の範囲】

【請求項１】　オブジェクト指向に基づく分散システムにおいて、

該オブジェクトを複数に複製して、複製されたオブジェクトの１つを実行系、残りを待機系とし、

各ノードのプラットフォーム毎に高信頼化・負荷分散機構を設け、オブジェクトとしての実体は存在しないが、オブジェクトの名前、識別子のみが前記高信頼化・負荷分散機構中に存在することで、外部から存在しているように観測される擬似管理オブジェクトが、該複製オブジェクトが存在するノードとは関係なく設置されるように設定され、該擬似管理オブジェクトと前記複製オブジェクトとで高信頼オブジェクトを形成し、該擬似管理オブジェクトが存在するノードのプラットフォーム内に、オブジェクトの名前、オブジェクト識別子を格納して高信頼化・負荷分散機構を設けるとともに、外部からはオブジェクトの存在が観測されるようにし、

該高信頼化・負荷分散機構は、該擬似管理オブジェクト宛に送信されたメッセージを検出した後、前記実行系複製オブジェクトに該メッセージを転送して、該複製オブジェクトのメソッドを起動させ、該オブジェクトの目的の動作を実行させ、

メソッドの実行の終了時に、実行系複製オブジェクトから待機系複製オブジェクトに対して内部状態の一貫性を維持するメッセージを送信することにより、複製オブジェクト間の内部状態の一貫性を保証し、

実行系複製オブジェクトが故障した時、前記擬似管理オブジェクトが存在するノード上の高信頼化・負荷分散機構と、待機系複製オブジェクトが存在するノード上の高信頼化・負荷分散機構との協調により、前記故障を検出し、

次に、複製オブジェクトの１つを選択して、該複製オブジェクトを実行系複製オブジェクトとし、故障以前の実行系複製オブジェクトが受け取ったメッセージを再現して、故障検出後の実行系複製オブジェクトに該メッセージを転送することにより、前記故障を回復し、

前記故障の検出後に新たに選択された実行系オブジェクトが、該オブジェクト以外の外部オブジェクトに作用する場合、該外部オブジェクトが２回目以降の作用であるときには再実行を妨げ、

実行系複製オブジェクトに故障が発生しない場合、該実行系複製オブジェクトの１メソッド実行終了毎に複製管理に関する情報中の不要な情報を無効化し、

ノード間で複製管理に関する情報を通信する場合、該ノードの負荷に関する情報を同時に送信し、該情報を利用して負荷分散を実行することを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項２】　請求項１に記載の分散システムにおける高信頼化と負荷分散方法において、

前記複製オブジェクトの内部状態の一貫性を維持するメッセージは、実行系複製オブジェクトの１メソッドの実行終了毎のチェックポイントとして、待機系複製オブジェクトが存在するノードの高信頼化・負荷分散機構に実行系複製オブジェクトの内部状態に関する情報を通知し、

該情報を受け取った前記高信頼化・負荷分散機構が待機系複製オブジェクトの内部状態を更新することを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項３】　請求項１に記載の分散システムにおける高信頼化と負荷分散方法において、

前記故障検出の手順は、前記擬似管理オブジェクトが存在するノードの高信頼化・負荷分散機構が外部から送信されてきた該擬似管理オブジェクト宛のメッセージを検出し、

該メッセージを実行系の複製オブジェクトに送信し、該実行系複製オブジェクトをメソッド実行状態にし、

同時に、待機系複製オブジェクトが存在するノードの高信頼化・負荷分散機構に、該メッセージの複製を送信し、

該メッセージを受信した高信頼化・負荷分散機構が、該メッセージの受信を契機として、そのノードローカルなタイマで時間の測定を開始し、予め定めた時間内に実行系のメソッドの実行終了時に待機系複製オブジェクトのノードに送信される内部状態に関する情報が到着しない場合、実行系複製オブジェクトが故障していると判断することを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項４】　請求項１に記載の分散システムにおける高信頼化と負荷分散方法において、

前記故障回復の手順は、請求項３に記載の動作により故障を検出した場合、実行系を切り替えるために、全ての複製オブジェクトに実行系複製オブジェクトを最も高い優先度とする優先度を順番につけ、

待機系複製オブジェクトが存在するノードの高信頼化・負荷分散機構のうち、該ノードに存在し、かつ実行系の次の優先度を持つ複製オブジェクトを実行系とし、

擬似管理オブジェクトが存在するノードの高信頼化・負荷分散機構から、請求項３に記載されたように受け取っていたメッセージの複製を、該ノードの待機系であった複製オブジェクトに送信し、

さらに、他の複製オブジェクトが存在するノードの高信頼化・負荷分散機構と擬似管理オブジェクトが存在するノードの高信頼化・負荷分散機構に対して、実行系が切り替わったことを示すメッセージを送信し、

該擬似管理オブジェクトが存在するノードの高信頼化・負荷分散機構で持っている、実行系がどの複製オブジェクトであるかを管理している情報を変更し、

新たに実行系に切り替わった複製オブジェクトの存在するノード上の高信頼化・負荷分散機構が、以前実行系であった複製オブジェクトの存在するノード上の高信頼化

2

・負荷分散機構に対して、オブジェクトやノードの再起動のメッセージを送信し、複製オブジェクトの中で優先度が次に高いものが存在するノードの高信頼化・負荷分散機構が、故障検出後、予め定めた時間内に実行系が切り替わったことが通知されない場合、実行系に切り替わり、上記の回復処理を代わりに行うことを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項５】　請求項１に記載の分散システムにおける高信頼化と負荷分散方法において、

前記外部のオブジェクトへの２回目以降の作用を防ぐ手順は、

故障回復時に、異なる複製オブジェクトとはいえ、１つの高信頼オブジェクトの中で、同じメソッドが実行されることによる外部への影響を防ぐために、

請求項３に記載の擬似管理オブジェクトが存在するノードの高信頼化・負荷分散機構が、複製オブジェクトに送信するメッセージの中に擬似管理オブジェクトのオブジェクト識別子と外部からメッセージを受信した時のそのノードにおけるローカルな物理的もしくは論理的時刻をつけ、

実行系複製オブジェクトが、外部へのメッセージ送信、もしくは実行環境に依頼する処理で外部へ作用を与える場合、

該メッセージ、もしくは実行環境に依頼する処理の関数の引数に、前記擬似管理オブジェクトのオブジェクト識別子と擬似管理オブジェクトが存在するノードローカルな時刻を付与し、これらをまとめて再実行防止タグと定義し、

該再実行防止タグつきのメッセージを受信した、もしくはタグつきの処理の依頼があった場合には、該タグとそれらによる処理の最終結果を組にして高信頼化・負荷分散機構が保存し、

再実行時に、該タグと同じものがあると検出された場合には、再実行を行うことなく、最終結果が必要である場合には、保存されていた値を返送することにより複製オブジェクトの再実行の副作用を抑え、

それにより、チェックポイントから選択すべきロールバックポイントを直近のメソッドの終了時点とすることを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項６】　請求項１に記載の分散システムにおける高信頼化と負荷分散方法において、

前記複製管理に関する情報で不要となるものを無効化する手順は、

複製を管理する情報で不要になったものを破棄するために、請求項２に記載のあるメソッドの実行の終了時に、実行系の複製オブジェクトから待機系複製オブジェクトに内部状態に関する情報が通知される時に、請求項５に記載のメソッド内で行われた外部へのメッセージ送信と実行環境依頼処理に関して、タグとともに保存してある情報を無効化するメッセージを、実行系の複製オブジェクトが存在するノードの高信頼化・負荷分散機構が、作用された側のノードの高信頼化・負荷分散機構に対して送信することにより、不要となった情報を無効化することを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項７】　請求項１に記載の分散システムにおける高信頼化と負荷分散方法において、

前記高信頼化・負荷分散機構では、カーネル内のメッセージハンドリング機構が、擬似オブジェクトの識別子から複製オブジェクトの識別子に変換する少量・高速の変換テーブルを持ち、実行系の複製オブジェクトにメッセージを転送し、高信頼オブジェクトの実行系の複製オブジェクトのメソッドの終了を検出し、待機系複製オブジェクトが存在するノードの複製メッセージ管理サーバオブジェクトにメソッド実行終了を通知し、

複製メッセージ管理サーバオブジェクトが、待機系の複製オブジェクトが存在するノードにおいて、実行系の複製オブジェクトが受信したメッセージの複製を保存し、

実行系の複製オブジェクトが故障なくメソッドの実行を終了した通知を受信したならば、保存しているメッセージを破棄し、さらに該実行の終了通知が予め定めた時間内に来ないことにより実行系の故障を検出し、さらに故障検出時には、他のノードに対しても回復処理の指示を出し、

複製オブジェクト識別子管理サーバオブジェクトが、カーネル内の擬似オブジェクトの識別子から複製オブジェクトの識別子へ変換するテーブルが不足した時に、カーネル内メッセージハンドリング機構からの問い合わせにより、複製オブジェクトの識別子を返し、

オブジェクト実行結果管理サーバオブジェクトが、あるオブジェクトがメソッドの実行を終了した時に、該メソッドの起動の契機となったメッセージと実行の結果の保存を行い、故障時に、同じ高信頼オブジェクトからのメッセージ再送であることを検出し、同じ動作を行うことを防ぎ、

オブジェクト複製オブジェクト実行管理サーバオブジェクトが、あるノード上の複製オブジェクトの実行の状態や高信頼化の優先度についての情報を管理し、複製オブジェクトの抹消や起動を行うことを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項８】　請求項１に記載の分散システムにおける高信頼化と負荷分散方法において、

前記ノード間の負荷分散の手順は、請求項２に記載の待機系複製オブジェクトが存在するノードの高信頼化・負荷分散機構に対して、実行系複製オブジェクトの内部状態に関する情報を通知する場合に、実行系複製オブジェクトが存在するノードの負荷を表す情報を通知し、

待機系複製オブジェクトが存在するノードの高信頼化・負荷分散機構は、前記情報を用いて自ノードの方が負荷

が軽いことを検出した場合には、擬似管理オブジェクトが存在するノードの高信頼化・負荷分散機構に実行系の切替要求を通知し、

該擬似管理オブジェクトが存在するノードの高信頼化・負荷分散機構は、通知された負荷に関する情報を用いて、複製オブジェクトのうちのどれを実行系とするかを選択することを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項9】 請求項1に記載の分散システムにおける高信頼化と負荷分散方法において、

前記ノード間の負荷分散の手順は、擬似管理オブジェクトが存在するノードにある高信頼化・負荷分散機構にノードの各時刻毎の負荷情報以外の情報を通知することにより、複製オブジェクトのうちのどれを実行系とするかを選択し、

動的に負荷分散を行うことを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項10】 請求項1に記載の分散システムにおける高信頼化と負荷分散方法において、

前記ノード間の負荷分散の手順は、請求項8に記載された擬似管理オブジェクトの存在するノードに集積されるノードの負荷の情報を用い、複製オブジェクトを負荷の軽いノードに移動することを特徴とする分散システムにおける高信頼化と負荷分散方法。

【請求項11】 請求項1に記載の分散システムにおける高信頼化と負荷分散方法において、

前記ノード間の負荷分散の手順は、請求項8に記載された擬似管理オブジェクトの存在するノードに集積されるノードの負荷の情報を用い、複製オブジェクトの生成場所と優先度を決めることを特徴とする分散システムにおける高信頼化と負荷分散方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、分散コンピューティング環境で動作するオブジェクトにおいて、複数の実行実体を実行系と待機系に分割することにより高信頼化と負荷分散とを実現することができる分散システムにおける高信頼化と負荷分散方法、ならびにその処理プログラムを記録した記録媒体に関する。

【0002】

【従来の技術】オブジェクト指向システムの動作単位であるオブジェクトを高信頼化する方法として、オブジェクトを分散コンピューティング環境で動作させる技術がある。分散システムで複製を用いて高信頼性を実現する場合、その複製の間で一貫性を維持する機構のオーバーヘッドが大きくなる。例えば、後述のPassive Replicationを用いる方法では、内部状態一貫性を維持するための特別な処理が必要となり、またActive Replicationを用いる方法でも、全ての複製が同じ順序で処理を行い、外部への作用は1つにまとめるという処理が必要になる。また、分散システムでは、故障の検出を効率よく行うことが困難であるため、多くのシステムでは時計を用いて故障を検出している。しかしながら、故障が生じていないにもかかわらず、通信網やノード自体の負荷のために、予想を越える時間を処理にかけた場合には、この方式では故障とみなされることがある。従来、実際に研究された例としては、国際会議“International Symposium on Fault－Tolerant Computing”において、“The Delta－4 Approach to Dependability in Open Distributed Computing System”（1988年）と、“Using Passive Replication in Delta－4 to Provide Dependable Distributed Computing”（1989年）に発表されたDelta－4というアーキテクチャがある。これは、Passive Replication／Active Replicationのいずれの場合についても研究されているが、故障検出に関する上述の問題は依然として解決されていない。

【0003】また、Active Replicationで、外部への作用を1つにまとめる処理を言語処理系でサポートするアプローチの例として、Fault－Tolerant Concurrent Cがある。これも、“International Symposium on Fault－Tolerant Computing”に、“Fault－Tolerant ConcurrentC：A Tool for Writing Fault－TolerantDistributed Programs”として発表されている。この言語処理系を用いて実行コードを得ると、複数の複製の外部への作用をカーネルが取りまとめるシステムコールも自動で追加され、Active Replicationを実現しているために、アプリケーションプログラマには複製の存在を意識させない。しかし、実行時のオーバーヘッドが非常に大きくなるため、Active Replicationを必要としない場合には、適用することはできない。さらに、Passive Replicationを用いる研究としては、“IEEE Transactions on Software Engineering”（1985年6月号）に掲載された“Implementing Fault－Tolerant Distributed Objects”がある。これは、基本的にはPassive Replicationであって、外部への作用を行わせない機構を考えている。しかし、あるFault－Tolerant Distributed Objectからの作用であることを判別するのに、実行スレッドの識別子を用いており、これが多段になると検出するため

に高コストとなる。通常、実行時には、実行実体の複製を作成せずに、必要データのみをノードに複製しておき、故障時になって実行実体を再生するＬａｓｙ　Ｆａｕｌｔ　Ｔｏｌｅｒａｎｃｅの考えに基づくｎＯＲのシステムも研究されている。これについては、『情報処理学会論文誌』（1995年10月号）の「データ分散化とオブジェクト再構築に基づく分散処理システムの高信頼化方式」として、発表されている。しかし、この方法であれば、故障回復からの時間が大きくかかるために、実時間性の要求が厳しいシステムには採用できない。

【０００４】

【発明が解決しようとする課題】このように、前述の各方式では、基本的に故障検出と内部状態の一貫性維持処理を別個に行っている。これでは、実行実体間等に余分なメッセージが飛び交い、性能の劣化を招くことになる。また、実行実体を複数作成する場合、高信頼化の他に負荷分散も行うことができるが、負荷分散のためには各ノードの負荷の情報が必要となる。この情報を集めるために、新たなメッセージが飛び交い、性能劣化の一原因となる。結局、下記問題点を解決することが本発明の課題となる。すなわち、従来の技術では、実行実体を複数設けて高信頼動作するオブジェクトを分散システム上に作成する場合に、複製間の内部状態の一貫性維持、故障検出、実行系の選択・切り替え、高信頼動作するオブジェクトの再起動による外部への作用をなくすことに余分な実行コストがかかるという点と、実行実体を複数持つために、それを用いた負荷分散の実現にも、余分な実行コストがかかるという点である。本発明の目的は、これら従来技術の課題を解決し、故障検出と内部状態一致と負荷分散を同時に行え、メッセージ数の少ない高信頼化・負荷分散機構を実現することができ、同時に複数の高信頼オブジェクトを実行する際にも妨げとならない、低コストの分散システムにおける高信頼化と負荷分散方法を提供することにある。

【０００５】

【課題を解決するための手段】上記目的を達成するため、本発明の分散システムにおける高信頼化と負荷分散方法では、タイムアウトによる故障検出のための時間測定開始要求メッセージと、故障時のオブジェクト再起動のための複製メッセージを共通にし、複製間の内部状態の一貫性維持のためのメッセージと、故障時には必要になるはずであった複製メッセージの破棄要求メッセージと、負荷分散に必要になる各ノードの負荷に関する情報の通知メッセージを共通にし、負荷に関する情報を自律的に適切なノードに集め、外部には存在が感じられる擬似管理オブジェクトの時計を用いることにより、複製の再実行を防ぎ、これにより高信頼実行と負荷分散実行を得る。また、高信頼実行と負荷分散実行を獲得するための処理プログラムを記録した記憶媒体を実現する。

【０００６】

【発明の実施の形態】以下、本発明の動作原理および実施例を、図面により詳細に説明する。

（動作原理）図９は、本発明が適用される複数ノードのオブジェクトによる交換機制御の図である。近年、社会経済活動の多様化、高度化に伴って、大量化・複雑化・広域化etcで発生する情報を、より正確に早く処理するため、計算機を用いた情報処理システムの重要性が高まっている。例えば、公衆データ通信網では、多数の利用者が多数の相手と通信できるよう交換処理機能が必要となる。また、多種多様な計算機や端末が公衆データ通信網に接続されるため、それらに対応する通信処理機能も必要となる。図９では、データ交換網内に、複数のノードが通信網（交換機）により接続され、各ノード内でドライバオブジェクト、ＩＮオブジェクトが設置されている。さらに、各ノード内のオブジェクト相互間でメッセージの送受信が行われる。図１０は、分散プラットフォーム（分散システム）の図である。複数のノード２５のオブジェクト１０８間でメッセージ通信を行っている場合に、プラットフォーム３０００より上の部分では分散システムを全く意識しないですむ。一方、プラットフォーム３０００より下の部分、つまりサーバ類、カーネルでは、分散システムを意識している。カーネル相互間では、通信網ハードウェアを介して実際の通信が行われる。

【０００７】ところで、このような情報処理システムの高信頼動作を実現するための方法として、一般に実行実体を複製する手法がある。この複製の実行形態としては、全ての複製が同時に目的とする機能を実行するＡｃｔｉｖｅ　Ｒｅｐｌｉｃａｔｉｏｎと、１つの複製のみが目的とする機能を実行し、残りは待機するＰａｓｓｉｖｅ　Ｒｅｐｌｉｃａｔｉｏｎとがある。Ａｃｔｉｖｅ　Ｒｅｐｌｉｃａｔｉｏｎは超高速な故障の回復が可能であるが、複製を外部に作用するものを集める必要があり、実行コストが高くなる。これに対して、Ｐａｓｓｉｖｅ　Ｒｅｐｌｉｃａｔｉｏｎは超高速な故障の回復はＡｃｔｉｖｅ　Ｒｅｐｌｉｃａｔｉｏｎに比べると困難であるが、分散システムで実行するのに適しており、かつ実行コストも低く抑えることができる。分散オブジェクト指向実行環境においてＰａｓｓｉｖｅ　Ｒｅｐｌｉｃａｔｉｏｎで高信頼化を獲得するのに、複製（以下、複製オブジェクト）の間では内部状態の一貫性を保証する必要がある。これを保証しないと、実行実体（以下、実行系）を切り替えたときに正しい実行を保証することができない。これを保証するため、内部状態を適当なチェックポイントのタイミングで破壊され難い場所に保存する。この場合の保存場所は、動作しているＣＰＵの近辺の例えばハードディスクのような場所でも、また遠隔地にある計算機のメモリ上でも差し支えはない。故障が起きた場合には、その保存されている時刻の集合から、関連するオブジェクトの間で最適な地点（ロールバック

ポイント）をそれぞれ決定して、そこから実行を再開することになる。このチェックポイントからロールバックポイントを決定するには、オブジェクトが互いにメッセージ送信等を介して外部に作用することがあるため一般に困難である。このことは、例えば情報処理学会誌『情報処理』（1993年11月号）の真鍋義文、青柳滋己共著『分散チェックポイント・ロールバックアルゴリズム』に解説がある。

【０００８】図７は、メッセージの複製がない場合に故障が起きた際の状況を示す図であり、図３は、高信頼オブジェクトのメッセージの受信を示す図であり、図５は、高信頼オブジェクトのメソッドの終了を示す図である。本発明においては、前記の問題を全ての高信頼オブジェクトの実行中の各メソッドの終了時点をチェックポイントとし、メッセージ等による外部への影響は再実行を防ぐ機構とすることにより、解決している。また、オブジェクトの再起動を行う場合には、オブジェクトの実行の契機となるメッセージが必要となる。図７に示すような方式、つまり実行系の複製オブジェクト１０５にのみメッセージ４２０を送信する方式であれば、ノード２０全体が故障した場合には、メッセージ４２０自体が喪失され、たとえ実行系が複製オブジェクト１０５に切り替わっても再起動は不可能である。すなわち、ノード２０に実行系複製オブジェクト１０５があり、ノード２１には待機系複製オブジェクト１０６が存在するにもかかわらず、オブジェクト１０５のみメッセージ４２０を送信したため、故障でメッセージ４２０が喪失され、ノード２１に切り替わってもメッセージ４２０がないため再実行できない事態になる。そのために、図３に示すようなメッセージの複製が必要となる。さらに、故障検出を分散システムで行うには、時計を用いて一定時間以上が経過するまでに返答がない時に、故障であると疑うタイムアウト方式を用いるのが一般的である。しかし、この時に、どの時刻から時間を測定するのか、あるいは何が起こるまで時間を計測するのか、をそれぞれ決定する必要がある。

【０００９】本発明においては、この問題を前述したメッセージの複製と内部状態の一貫性保証のためのメッセージを組み合わせて解決している。つまり、図３に示すように、複製されたメッセージが各ノード（１１，１２，１３）に分散され、これにより待機系のノード１２と１３で時間の計測を開始し、メソッド終了時に内部状態の一貫性保証のためのメッセージ（図５の４１２，４１３）が来るまでの時間を計る。これにより、故障検出のためだけに必要となる余分なメッセージはノード間に飛ぶことはない。さらに、擬似的な管理系（図３の擬似管理オブジェクト１１００のあるノード１０）のローカルな時刻と擬似管理オブジェクト１１００のオブジェクト識別子２０１を用いて、高信頼オブジェクトの中の実行系の複製オブジェクトの外へメッセージ等を介して行う作用の識別を行う。これにより、複製オブジェクトの再起動による同じ高信頼オブジェクトからの同じメッセージが外部に及ぼす影響をなくすことができる。これにより、故障時に実行系の複製オブジェクトが複数存在する瞬間があった場合にも対処することができる。そして、内部状態一貫性保証のメッセージ中に実行系のノード（図５のノード１１）の負荷状況を待機系のノード（同じくノード１２，１３）に通知することにより、待機系のノードが自律的に判断し、自ノード上の複製オブジェクトが実行系になるべきと判断した場合に、擬似管理系のノード（同ノード１０）に通知し、それにより負荷分散も同時に実現する。なお、図３、図５については、さらに詳述する。

【００１０】（実施例）

〔高信頼オブジェクトの構成〕図１は、本発明の一実施例を示すオブジェクトの高信頼化システムのソフト構成図である。図１において、実行系の複製オブジェクト１２００と待機系の複製オブジェクト１３００，１３０１から高信頼オブジェクト１０００は構成されている。例えば、外部のオブジェクト１００発のオブジェクト識別子２０１、オブジェクトの名前３０１宛のメッセージ４００がノード１０に到達すると、ノード１０の高信頼化・負荷分散機構（カーネル内のメッセージハンドリング機能）がそのメッセージを検出して、複製オブジェクト１２００へ転送する。メッセージ４００の転送を受けた複製オブジェクト１２００は、メソッドを実行する。この時、外部から観測すると、送信したメッセージは何のエラーも起こすことなく処理されているので、オブジェクト識別子２０１、オブジェクトの名前３０１のオブジェクトが存在しているように観測される。しかも、複製オブジェクトの存在は観測されない。逆に、実行系の複製オブジェクト１２００からのメッセージは、高信頼化・負荷分散機構の処理によりノード１０上のオブジェクト識別子２０１、オブジェクトの名前３０１のオブジェクトから送信されているメッセージ４０１であるかのように観測される。この時も、外部からはオブジェクト識別子２０１、オブジェクトの名前３０１のオブジェクトが存在しているように観測される。つまり、ノード１０と１２の高信頼化・負荷分散機構のメッセージの処理機構により、実際には存在しない擬似管理オブジェクト１１００が存在し、その擬似管理オブジェクト１１００が複製オブジェクト１２００，１３００，１３０１の動作を管理しているものと把握することができる。外部からのアクセスは、擬似管理オブジェクト１１００に対して行っているように観測される。

【００１１】〔高信頼化・負荷分散機構の構成〕図２は、高信頼化・負荷分散機構のソフト構成図である。なお、各サーバオブジェクトの詳細は後述する。図２において、高信頼化・負荷分散機構２０１０はノード１６のプラットフォーム３００１内に格納されており、これが

ノード１７，１８，１９に複製され、転送されて格納される（それぞれ２０１１，２０１２，２０１３）。カーネル２１００は、通常の分散ＯＳのカーネルに特定のオブジェクトに関するメッセージの検出機能が付加されたものである。通常のオブジェクト間通信４２２は、ノード１７とノード１８のオブジェクト間で行われているが、実際にはプラットフォーム３００１内に下りてから目的オブジェクトに振り分けられる（４２３，４２４，４２５，４２６）。ノードを跨る場合、例えばノード１９を跨る場合には、メッセージキュー（２０５０，２０６０）を経由する。そのために、カーネル２１００のスケジューラ・メッセージハンドラ等に少量の改造を加えることにより、この検出機能が実現される。この検出機能により、擬似管理オブジェクト宛のメッセージと複製オブジェクトの外部への作用を検出し、高信頼動作・負荷分散動作に関係する処理を行う。カーネルにより、擬似管理オブジェクトから実行移の複製オブジェクトへメッセージが送信される時に、そのメッセージの複製が作られる。この複製されたメッセージは、待機系の複製オブジェクトの存在するノードの複製メッセージ管理サーバオブジェクト２２００によって受信・保存される。また、複製メッセージ管理サーバオブジェクト２２００は、このメッセージ受信から時間を計り、故障検出にも関係する。

【００１２】同じ高信頼オブジェクトの中の異なる複製オブジェクトからのメッセージ送信等が起きる時に、そのメッセージ送信によるメソッド起動を防ぐ機構をオブジェクト実行結果管理サーバオブジェクト２３００により実現する。複製オブジェクト識別子管理サーバオブジェクト２４００は、カーネルが擬似管理オブジェクトのオブジェクト識別子等から実行系の複製オブジェクトの識別子等を得る時に、カーネルが使用するメモリ領域を小さくするために用いられる。具体的には、カーネル２１００の中の変換テーブルに目的とするエントリがない場合には、複製オブジェクト識別子管理サーバオブジェクト２４００に問い合わせる。また、外部から複製オブジェクト識別子管理サーバオブジェクト２４００にメッセージを送ることにより、カーネルの中の変換テーブルの内容を変更することもできる。複製オブジェクト実行管理サーバオブジェクト２５００は、そのノード上に存在している複製オブジェクトの実行状態を管理する。外部からの問い合わせにより、実行状態を返したり、オブジェクトを末消・再起動する機能も持っている。上記サーバオブジェクト（２２００～２５００）は、各ノードに１つ決まっているので、オブジェクト識別子はノード番号から容易に分かる識別子を割り当てる。こうすることにより、後述の高信頼動作・負荷分散動作を行う時に、オブジェクトの名前からオブジェクトの識別子を得る等の操作が不要となる。

【００１３】〔高信頼オブジェクトのメッセージ受信〕図３は、高信頼オブジェクトがメッセージを受信して、メソッドを起動する状態を示す図である。高信頼オブジェクトは、外部から観測すると、図１に示すように擬似管理オブジェクトが存在する場所に存在しているように見える。外部からのメッセージ４０５は、擬似管理オブジェクト１１００が存在するノード１０に到着すると、ノード１０の高信頼化・負荷分散機構２０００（カーネル内のメッセージハンドリング機構）が擬似管理オブジェクト１１００宛であることを検出する。カーネル内に備えられたテーブルにより、擬似管理オブジェクト１１００から実行系の複製オブジェクト１２００と待機系の複製オブジェクト１３００，１３０１とそれぞれの優先度を得る。そのテーブル上に求めるデータが存在していない場合には、ノード１０の中の高信頼化・負荷分散機構２０００の複製オブジェクト識別子管理サーバオブジェクトに問い合わせて、必要なデータを得る。ここで、複製オブジェクトの優先度は、任意の順序で付与すれば本発明の動作を行うことができるが、擬似管理オブジェクト１１００の存在するノード１０とメッセージ通信が早く行えるノード上の複製オブジェクトに高い優先度を与えれば、実行効率を高めることができる。ノード間のメッセージ通信の早さに差が生じるのは、通信路の混み具合も各ノード間で異なり、ノード自体の負荷も異なるからである。高信頼化・負荷分散機構２０００（カーネル内メッセージハンドリング機能）は、受け取ったメッセージ４０５を実行系の複製オブジェクト１２００に送信する（４０６）。同時に、待機系の複製オブジェクトの存在しているノード１２，１３の高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）に対しても、メッセージを複製して送信する（４０７，４０８）。後の複製されたメッセージ（４０７，４０８）には、ノード１０のローカルタイマで測定したメッセージ４０５の受信の時刻を入れる。この時刻が高信頼オブジェクト１０００のメソッド起動時刻になる。なお、この時刻は、実世界の時刻を示す物理的時刻でも、コンピュータシステム中での論理的時刻でも構わない。分散システムでの論理的時刻としては、メッセージ中に含み、メッセージ受信時には、自分の論理時刻とメッセージ中の論理時刻のうち大きい方に１を加えたものを新しい論理時刻とするＶｅｃｔｏｒ　Ｃｌｏｃｋが例として挙げられる。待機系の複製オブジェクトが存在するノード１２，１３では、到着したメッセージ４０７，４０８は高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）が保存するので、複製オブジェクト１３００，１３０１は動作しない。高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）は、メッセージ４０７，４０８を受信すると、ノードローカルなタイマで時間の測定を開始する。これは、後述する故障検出に用いる。

【００１４】〔高信頼オブジェクトのメソッドの実行〕図４は、高信頼オブジェクトの実行の状態を示す図である。実行系の複製オブジェクト１２００は、外部への作用以外には、複製オブジェクトが通常のオブジェクトとして動作する。オブジェクト１２００がオブジェクト外部へ作用する場合には、メッセージの送信を用いる。故障発生時に実行系を切り替え、再実行するのは、メソッドの先頭からとなる。従って、あるメソッド５００で、外部のオブジェクト１０１へのメッセージ送信４１０があるならば、故障時に、複製オブジェクトの再実行を行うと、オブジェクト１０１へメッセージ４１０と同じ内容のメッセージの送信を行うことになる。これでは、オブジェクト１０１の状態が余分なメッセージを受ける取ことにより、目標とする正しい状態とは離脱することになる。例えば、オブジェクト１０１が内部状態を受けたメッセージの数で変化する場合や、外部に新しいオブジェクトを生成する場合には、不変なメッセージが故障により再送信されることによる副作用は抑えなければならない。すなわち、故障時に高信頼オブジェクト１０００がメソッド５００を２回以上実行することがあっても、外部のオブジェクト１０１への作用を起こさないような機構が必要である。高信頼オブジェクト１０００の実行系複製オブジェクト１２００からのメッセージ４１０には、高信頼オブジェクトのオブジェクト識別子、擬似管理オブジェクトでのメッセージ受信時刻、メソッド識別子、そのメソッド内での外部への作用についての逐次カウンタの値をタグ１５００としてつける。

【００１５】通常の高信頼動作・負荷分散動作に関係のないメッセージにもタグはついているが、メッセージ中にフィールドとして存在しているだけで、値としては有効ではない。あるノード１５が、自ノードのオブジェクト１０１宛のメッセージ４１０を受信した場合、その高信頼に関するタグ１５００が有効であるか否かを調べる。有効であれば、そのタグ１５００と同じ内容のものがないか否かを調べる。もしなければ、それを保存して、メソッド５０１の実行を開始する。そして、メソッド５０１の実行が終了した場合には、その最終結果とタグ１５００を合わせて保存する。もし、タグ１５００が有効であり、かつタグ１５００と同じ内容のものが存在していた場合には、同じ高信頼オブジェクト１０００の中で、故障により実行系複製オブジェクトが切り替わったものと判断して、メソッド５０１の実行は行わない。さらに、保存しておいた最終結果を高信頼オブジェクトへ返すような返答つきメッセージ送信であった場合には、保存してあった最終結果を返答とする。これらの保存してある情報の無効化を行うべき時には、高信頼オブジェクトのメソッドの実行の終了時であり、これについては後述する。

【００１６】〔高信頼オブジェクトのメソッドの実行の終了〕図５は、高信頼オブジェクトのメソッドの実行の終了動作を示す図である。実行系複製オブジェクト１２００のメソッド５００の実行の終了をそのノードの高信頼化・負荷分散機構が検出し、高信頼化・負荷分散機構２００１にその終了を通知する。この時点で、外部のアプリケーションオブジェクトからは、高信頼オブジェクト１０００のメソッド５００の実行は終了しているように見受けられる。その終了通知を受けた高信頼化・負荷分散機構２００１は、メソッド５００で行われた外部へのメッセージ送信、もしくは実行環境依頼処理の履歴を調べ、それが存在していたならば、その送信先ノード１５の高信頼化・負荷分散機構（オブジェクト実行結果管理サーバオブジェクト）２００５に前述のタグ１５００と保持情報の無効化のメッセージ４１１を送信する。また、待機系の複製オブジェクトの存在するノード１２，１３の高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）に、メソッド５００を起動する契機となったメッセージ（図３の４０７，４０８）の破棄要求と、ノード１０の負荷に関する情報と、実行系複製オブジェクト１２００の内部状態に関する情報を含むメッセージ４１２，４１３を送信する。

【００１７】ノード１５がメッセージ４１１を受信すると、それについている高信頼化に関するタグ１５００が有効か否かを調べる。もし、有効であり、かつ内容が保持情報の無効化であれば、そのタグに対応するメソッドの実行結果を破棄する。ノード１２，１３にある高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）がメッセージ４１２，４１３を受信すると、擬似管理オブジェクト１１００の存在するノード１０の高信頼化・負荷分散機構２０００（カーネルのメッセージ等のリング機能）から送信されたメッセージ（図３の４０７，４０８）の中から、メソッド５００に対応するメッセージを破棄する。さらに、受け取った内部状態に関する情報で、待機系の複製オブジェクト１３００，１３０１の内部状態を更新する。さらに、実行系複製オブジェクトのノード１１に関する負荷の情報と自分のノードの負荷の情報を比較して、自分のノードの方が負荷が軽ければ、その負荷の情報を擬似管理オブジェクトの存在するノード１０の高信頼化・負荷分散機構２０００（複製オブジェクト識別子管理サーバオブジェクト）に通知する。

【００１８】〔複製オブジェクトの故障検出〕図６は、タイムアウトを用いた故障検出を示す図である。外部から高信頼オブジェクト１０００へ送信されたメッセージ４０５は、実際には擬似管理オブジェクトの高信頼化・負荷分散機構２０００（カーネルのメッセージハンドリング機能）が高信頼オブジェクト１０００宛であると検出し、実行系の複製オブジェクト１２００と、待機系の複製オブジェクト１３００，１３０１の存在するノード１２，１３の高信頼化・負荷分散機構２００２，２００

３（複製メッセージ管理サーバオブジェクト）に送信する。ノード１２，１３の高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）は、そのメッセージ４０７，４０８を蓄える。同時に、カーネル内のノードローカルなタイマで時間測定の開始を行う。実行系複製オブジェクト１２００のメソッド５００の実行が正しく終了した場合には、ノード１２，１３の高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）にその旨のメッセージ４１２，４１３が返って来るはずである。これが予め決められた時間内に終了しなければ、故障と判定する。

【００１９】〔故障検出による複製オブジェクトの切替え〕前述のように故障を検出した高信頼化・負荷分散機構２００２，２００３（複製メッセージ管理サーバオブジェクト）は、自分のノード上で動作している複製オブジェクト１３００，１３０１の優先順位を調べて、自分の方が高ければ、自分のノード上で動作している複製オブジェクトが実行系になると判断する。ここでは、複製オブジェクト１３００の方が優先度が高いとする。新たに実行系になった複製オブジェクト１３００は、自分のノードの高信頼化・負荷分散機構２００２（複製メッセージ管理サーバオブジェクト）から保存されているメッセージ４０７を受け取り、メソッドを実行する。高信頼化・負荷分散機構２００２（複製オブジェクト実行管理サーバオブジェクト）は、複製オブジェクト１３００の優先度を１つ上げる。次に、高信頼化・負荷分散機構２００２（複製メッセージ管理サーバオブジェクト）は、擬似管理オブジェクト１１００のあるノード１０の高信頼化・負荷分散機構２０００（複製オブジェクト識別子管理サーバオブジェクト）と他の待機系の複製オブジェクトのあるノード１３の高信頼化・負荷分散機構２００３（複製オブジェクト識別子管理サーバオブジェクト）に対して、実行系の切り替わりを通知する。高信頼化・負荷分散機構２００３（複製オブジェクト実行管理サーバオブジェクト）は、自分のノード上の複製オブジェクト１３０１の優先度を１つ上げる。

【００２０】さらに、以前実行系の複製オブジェクトが存在していたノードの高信頼化・負荷分散機構２００１（複製オブジェクト実行管理サーバオブジェクト）に対して、高信頼化・負荷分散機構２００２（複製オブジェクト実行管理サーバオブジェクト）が複製オブジェクト再起動の要求を通知する。このメッセージが通知されると、ノード１１の複製オブジェクト１２００が再起動される。この時に、複製オブジェクト１２００がまだ存在していたならば、抹消の後、再起動される。ノード１１で複製オブジェクトが再起動された場合には、内部状態は一貫性を保っていない。しかし、あるメソッドの実行が終了した時には、実行系の複製オブジェクトは待機系の複製オブジェクトに対して内部状態の一貫性を保証するメッセージを送信する。従って、故障時に待機系の複製オブジェクト再起動が発生しても、内部状態の一貫性を保証する処理は新たに必要としない。たとえ、ノード１１の負荷が異常に高いこと等により、複製オブジェクト１２００の反応が遅く、故障していないにもかかわらず故障していると判断され、上記の回復処理が進んで元の実行系の複製オブジェクト１２００と新しい実行系の複製オブジェクト１３００が存在しても、オブジェクト実行結果管理サーバオブジェクトの作用により、外部への影響は出ない。優先順位の低い待機系複製オブジェクトのあるノード１３の高信頼化・負荷分散機構２００３（複製メッセージ管理サーバオブジェクト）が故障を検出してから、一定時間以内に実行系の切り替わりが生じなければ、自分より優先度の高い待機系複製オブジェクトが全て故障していると判断して、自分が実行系になるための上記と同様の処理を行う。これらのテーブルの更新が行われる時に、カーネル内のテーブルに載せることができなくなったデータの内容は、複製オブジェクト識別子管理サーバオブジェクトへ通知される。

【００２１】〔複製オブジェクトの負荷による切替え〕先に、図５の説明により高信頼オブジェクト１０００におけるメソッド５００の実行の終了の状態を示した。そこで、実行系の複製オブジェクトがメソッドを終了した時に、高信頼化・負荷分散機構２００１，２００２，２００３の動作により各ノード１１，１２，１３の負荷の情報が高信頼化・負荷分散機構２０００（複製オブジェクト識別子管理サーバオブジェクト）に集積する機構について説明した。これにより、擬似管理オブジェクトの存在するノード１０の高信頼化・負荷分散機構２０００（複製オブジェクト識別子管理サーバオブジェクト）は、どの複製オブジェクトが、負荷が軽いノードで実行されているのかを判別し、次回以降にどの複製オブジェクトを実行系とすべきかを決定する。これにより、高信頼化と負荷分散を同時に実現することができる。

【００２２】〔複製オブジェクトの外部情報による切替え〕前述のように、前回のメソッドの実行時のノードの負荷情報だけでなく、高信頼化・負荷分散機構２０００（複製オブジェクト識別子管理サーバオブジェクト）にノード選択の情報を外部から直接与えることにより、実行系の選択方法を変更することができる。具体的には、高信頼化・負荷分散機構２０００の中のカーネル内の複製オブジェクトに関するテーブルと複製オブジェクト識別子管理サーバオブジェクトの持つテーブルを、複製オブジェクト識別子管理サーバオブジェクトにメッセージを送信することにより、変更する。これにより、メソッドの実行ということを用いなくても、より柔軟な実行系の選択を行うことができる。

【００２３】〔オブジェクトの移動が可能な場合の負荷分散〕図８は、オブジェクト指向分散システムにおいて、オブジェクトを移動する場合の負荷分散システムの

構成図である。図８においては、擬似管理オブジェクト１１００と実行系複製オブジェクト１２００と待機系複製オブジェクト１３００，１３０１で１つの高信頼オブジェクトを構成しており、同時に、擬似管理オブジェクト１１０１と実行系複製オブジェクト１２０１と待機系複製オブジェクト１３０２，１３０３で１つの高信頼オブジェクトを構成している。ノード１０の高信頼化・負荷分散機構２０００（複製オブジェクト識別子管理サーバオブジェクト）には、ノード１１，１２，１３の負荷の情報と、ノード２２，２３，２４の負荷の情報が集積されている。本発明における外部情報を与える負荷分散方式においては、その外部情報をこの集積された負荷に関する情報とする（請求項９参照）。例えば、ノード２２，２３，２４の負荷が高くなり、故障がなくても実行系複製オブジェクト１２０１の処理が遅くなり、ノード２３，２４等により故障と診断された場合に、ノード１１，１２，１３の負荷が低い時には、複製オブジェクト１２０１，１３０２，１３０３をノード１１，１２，１３に移動して、全体のスループットを上げることも可能である。

【００２４】〔集積された負荷情報の利用方法〕本発明により、擬似管理オブジェクトの存在するノードには、特にノードの負荷を集めるためだけのメッセージ通信を行うことなく、周辺ノードの負荷に関する情報が集積されていくので、高信頼オブジェクトを生成する時に、複製オブジェクトをどこのノードに生成すべきか、また複製オブジェクトの優先度はいかにするか、という判断に適用することが可能である。

【００２５】〔Ｃａｌｌ　Ｍａｎａｇｅｒへの適用〕この他に、分散システムにおける呼処理プログラム中に存在するＣａｌｌ　Ｍａｎａｇｅｒと呼ばれるオブジェクトも適用対象である。Ｃａｌｌ　Ｍａｎａｇｅｒは、ある加入者の発呼を検出すると、実際に呼処理を行うＣａｌｌｅｒや受け側の呼処理を行うＣａｌｌｅｅを生成する。実行実体を複数持ち、適切に負荷の軽いところに処理を依頼できるので、全体のスループットは上がる。負荷分散の例では、実行系の選択にノードの負荷による方式（請求項８参照）と、外部情報による方式（請求項９参照）を共用することにより、以下のことが可能となる。すなわち、広域分散システムで、例えば、人間世界の時差により特定の地理的な場所におけるノードの負荷が高まるような場合、それは経験的に予測できるので、オペレーションシステムが高信頼オブジェクトに向けてその情報を与えればよい。このようにすれば、ある地域で時間的に暇なノードのＣＰＵ資源を借りることができ、全体のスループットを上げることが可能である。

【００２６】〔名前サーバの高信頼化・負荷分散への適用〕他の適用対象としては、分散システムにおける名前サーバ（オブジェクトの名前とオブジェクトの識別子を管理するサーバ）の高信頼化・負荷分散がある。名前サーバは、分散システム上でオブジェクトが通信するのには必須なサーバである。本発明により、名前サーバを構成すれば、名前に関する問い合わせに来るクライアントは実際にはどこの複製オブジェクトに依頼しているのかは完全に隠蔽され、また、高信頼化・負荷分散性も実現されているので、運用面でも問題がない。

【００２７】〔分散システムの高信頼化と負荷分散方法の処理プログラムを記録する記録媒体〕以上に述べたオブジェクト指向に基づく高信頼化と負荷分散方法は、高信頼化及び負荷分散を行うための処理プログラムとして実現可能であり、その処理プログラムは記録媒体に記録して提供することができる。

【００２８】

【発明の効果】以上説明したように、本発明によれば、故障検出と内部状態一致と負荷分散を同時に行うメッセージ数の少ない高信頼化・負荷分散機構の実現が可能となる。そして、本来ならば、通常のオブジェクトの実行と並行して故障検出のメッセージが飛び交うが、別のノードでのメッセージ保存と内部状態一致のためのメッセージにより、故障検出を行う。また、内部状態一致のためのメッセージ数に負荷情報を加え、負荷分散を行うことも可能である。これらにより、メッセージ数を大きく削減することができる。また、本発明では、高信頼化・負荷分散機構を複数サーバで実現するので、同時に複数の高信頼オブジェクトを実行する際にも妨げとならない。さらに、本発明によれば、故障検出と回復の機構についても、高信頼オブジェクトの外部への作用を抑える機構により、不要な待ち合わせを削除することができる。つまり、通常のタイムアウトによる故障検出であれば、一定時間内に返答がなければ故障と疑い、本当に故障であるか否かの確認の処理が必要となる。本発明では、一定時間内に返答がなければ、直ちに別の実行系を起動することができ、前の実行系の削除を依頼する。削除が行われる前に、前の実行系が同時に何か外部に作用を行っていても、それは影響がない。

【図面の簡単な説明】

【図１】本発明の一実施例を示すオブジェクト複製を予備系とする高信頼オブジェクトの図である。

【図２】本発明における高信頼化・負荷分散機構をカーネルとサーバオブジェクトで実現した構成図である。

【図３】本発明における高信頼オブジェクトのメッセージの受信を示す図である。

【図４】本発明における高信頼オブジェクトのメソッドの実行を示す図である。

【図５】本発明における高信頼オブジェクトのメソッドの終了を示す図である。

【図６】本発明における複製オブジェクトの故障検出を示す図である。

【図７】従来における故障の発生状態と回復不能状況を示す図である。

【図８】本発明におけるオブジェクト移動による負荷分散の説明図である。

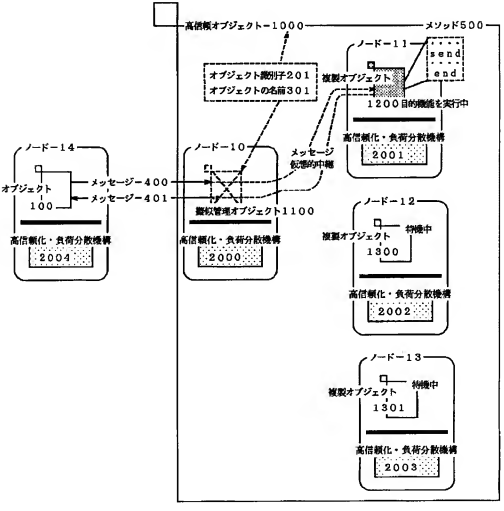
【図９】本発明が適用される複数ノードのオブジェクトによる交換機制御の図である。

【図１０】本発明が適用される分散プラットフォームの図である。

【符号の説明】

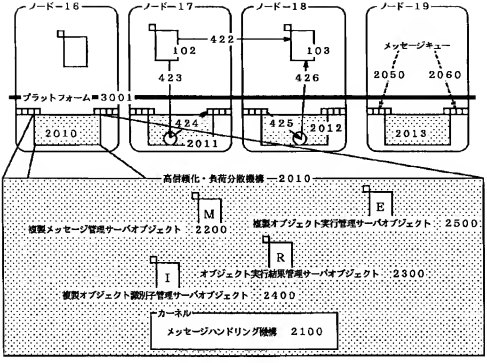
１０，１１，１２，１３，１４，１５，１６，１７，１８，１９，２０，２１，２２，２３，２４…ノード、１００，１０１，１０２，１０３，１０５，１０６…オブジェクト(一般)、２０１…オブジェクト識別子、３０１…オブジェクトの名前、４００，４０１，４２２，４２３，４２４，４２５，４２６，４０５，４０６，４０７，４０８，４１０，４１１，４１２，４１３，４１４，４１５，４２０…メッセージ、５００，５０１…メソッド、１０００…高信頼オブジェクト、１１００，１１０１…擬似管理オブジェクト、１２００，１２０１…実行系複製オブジェクト、１３００，１３０１，１３０２，１３０３…待機系複製オブジェクト、１５００…タグ、２０００，２００１，２００２，２００３，２００４，２００５，２００６，２０１０，２０１１，２０１２，２０１３…高信頼化・負荷分散機構、２０５０，２０６０…メッセージキュー、２１００…カーネル内メッセージハンドリング機構、２２００…複製メッセージ管理サーバオブジェクト、２３００…オブジェクト実行結果管理サーバオブジェクト、２４００…複製オブジェクト識別子管理サーバオブジェクト、３００１…プラットフォーム。

【図１】



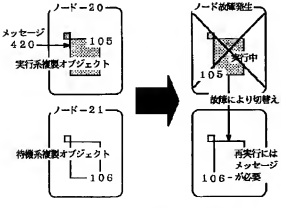

高信頼オブジェクトの構造

【図２】



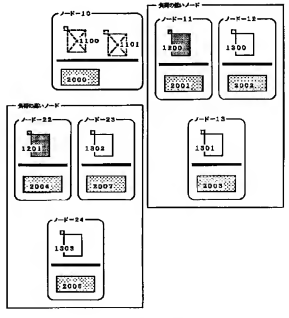

高信頼化・負荷分散機構

【図７】



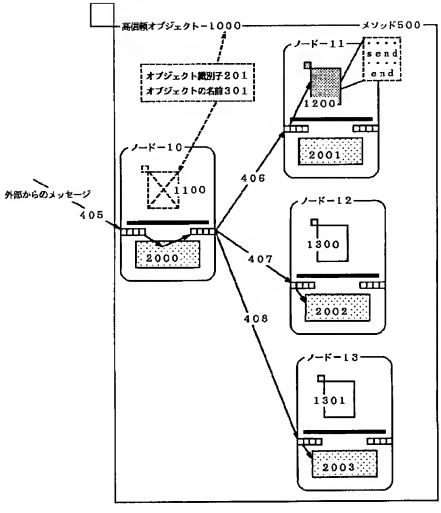

メッセージの複製がない場合

【図８】



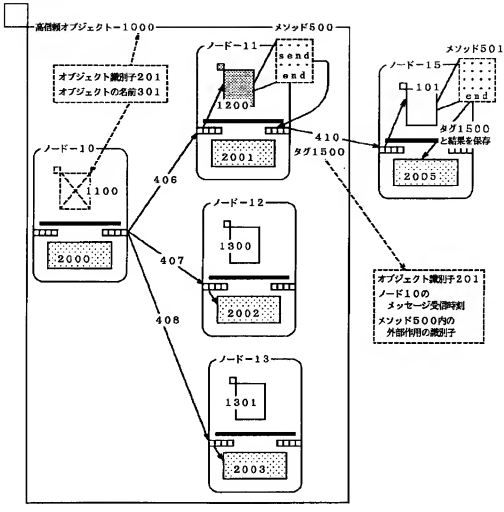

オブジェクト移動による負荷分散

【図３】



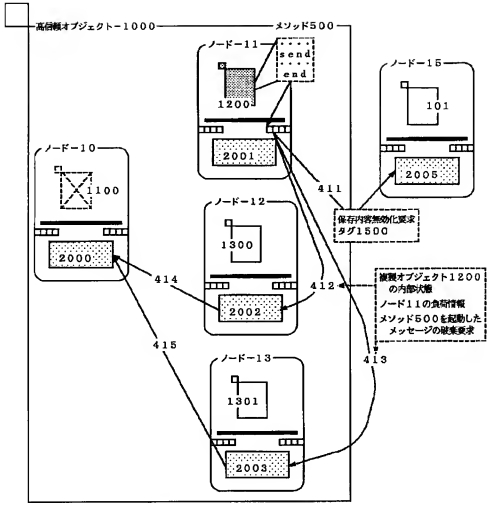

高信頼オブジェクトのメッセージ受信

【図４】



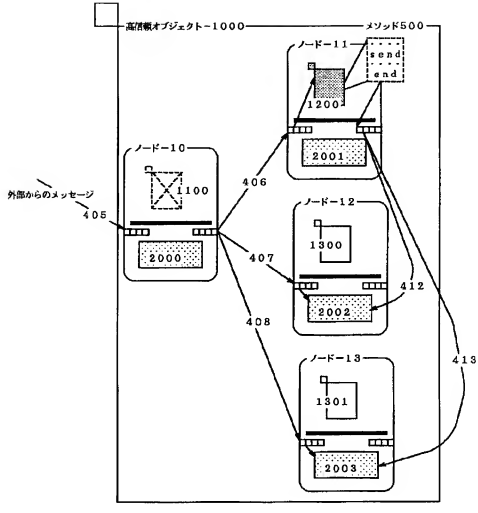

高信頼オブジェクトのメソッドの実行

【図５】



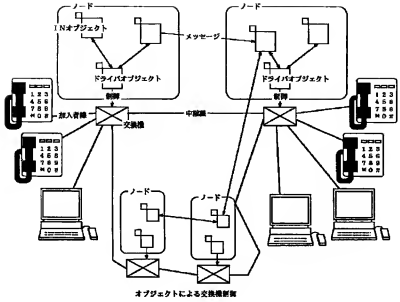

高信頼オブジェクトのメソッドの終了

【図６】

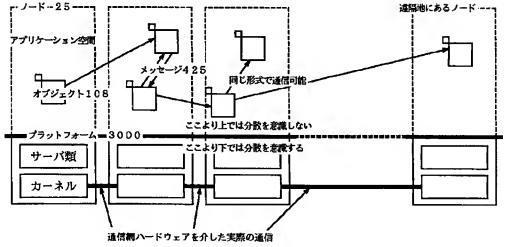



複製オブジェクトの故障検出

【図９】




【図１０】




分散プラットフォーム（分散システム）